

### REMARKS/ARGUMENTS

Claims 1-14 and 23-29 were rejected over the teachings of US Patent 5,692,183 (Hapner) either alone or in combination with US Patent 6,438,616 (Callsen), US Patent 5,692,183 (Snyder) and/or official notice taken by the Examiner.

The Examiner explained in paragraph 6 starting at the bottom of page 2 of the Office Action that Hapner described "creating a new object for a new instance of the application" in Hapner's abstract lines 12-15 and in column 5, lines 23-25. The Examiner further stated at the top of page 3 of the Office Action that Hapner described "using an existing object of an existing instance" at Hapner's column 8, lines 60-67 and column 9, lines 48-67. All of the just-described lines from Hapner's patent are reproduced below (including Examiner's emphasis):

An object developer develops object implementations which the distributed object ***generates distributed objects*** with, in the process automatically providing transparent persistence. Abstract, lines 12-15.

the object constructor being used to ***create an instance of an object*** in accordance with one embodiment of the present invention. Column 5, lines 23-25.

As background, an object "class" is a template from which an object can be created. It is used to specify the behavior and attributes common to all objects of the class. The mechanism by which ***new classes are defined from existing classes*** is "inheritance." "Subclasses" of a class inherit operations of their parent class. As is well known, "inheritance" is a mechanism by which reusability is facilitated. Column 8, lines 60-67.

The servant object 150 is one instance of a distributed object in accordance with one embodiment of the present invention. As will be appreciated by those skilled in the art of object oriented programming, when a new instance of a class is created, the memory for the instance variables is allocated. FIG. 3 diagrammatically illustrates the memory allocation of servant object 150. Transient data 152 is the memory portion of the servant object 150 which contains state (e.g. variables) which the object needs to use only as transient data. By way of analogy, the transient data may be envisioned as a scratch pad area for use only by this instance of the servant object 150. Data object 158 is the memory portion of the instance which the object developer intended for use as persistent state. This region of memory is pointed to by data object pointer 154, which is inherited from the above-described data object classes.

Initially, when the servant object 150 is created, an object constructor, such as object constructor 184, will create the servant object 150 with transient data 152 and data object 158 in an initialized state (which may be random values left... Column 9, lines 48-67.

Note that all of the above-quoted text refers to an **instance of an object**, in the art of object oriented programming. A typical object instance of the type referred to by Hapner is an instance of a datatype (class), such as the Java class "Integer" e.g. Integer i, j; wherein i and j are instances of the class (type) Integer. Hapner further describes his objects and instances at several locations, e.g. see column 1 lines 50-67, column 2 at lines 30-36, and column 3 at lines 35-48.

As will be apparent to the skilled artisan, an **instance of an application** as recited in Claim 1 is not the same as (and is therefore not disclosed or suggested by) an instance of an object in object oriented programming (OOP). For examples of application instances, see Applicants' originally-filed specification, at page 1 lines 23-26 which describes "multiple instances (e.g. Instance1 and Instance 2 in FIG. 1) of the database server". Also see the specification at page 2 at lines 21-23 which states that "a single instance of a database process (also called "Oracle instance") is executing on each of the computers ..."

However, in view of the Office Action, it appears that the Examiner has misunderstood the term "instance of the application" in originally filed Claim 1. Hence, Claim 1 is being clarified in this Amendment to state that each application instance includes multiple processes. Support for the just-described changes to Claim 1 can be found throughout Applicants' originally-filed specification, including, for example, page 2 line 24, and page 7 lines 7-18. In view of this clarification, Applicants respectfully submit that creating an object instance using Hapner's **object inheritance mechanism** of OOP fails to disclose or suggest creation of an application instance (which contains multiple processes).

Note further that that multiple instances of a single application as required in Claim 1 are nowhere disclosed or suggested by Hapner.

Applicants further submit that the above-described clarifications and related changes to Claim 1 do not narrow Claim 1, because Applicants have merely made explicit that which was inherently present in the claim as originally filed. Therefore there is no loss

of doctrine of equivalence in making these changes to Claim 1. If the Examiner believes that the scope of Claim 1 has changed, the Examiner is respectfully requested to explain their belief in the next Office Action.

In view of the above remarks, Applicants respectfully traverse the Examiner's anticipation rejection as being prima facie defective. Specifically, the Examiner explained (emphasis added) that Applicants' "creating a new object for a new instance ..." was being rejected (see bottom of page 2 of the Office Action), over Hapner's disclosure to generate distributed objects and to "create an instance of an object..."

Note that distributed objects generated by Hapner are within a "distributed object environment" (see Hapner's column 6 at lines 40-41 and column 8 at line 8) and/or an "object development framework" (see Hapner's column 8 at lines 46-47 and column 8 at lines 54-55). Hence, Hapner's object is being created via OOP object inheritance mechanism, within an application that itself must be up and running. In contrast, Claim 1 is describing the creation of an object for use by a new application instance that is yet to start execution. Therefore, Claim 1's method is performed outside of the application (which is being started), a concept nowhere disclosed or suggested by Hapner.

Applicants have further clarified Claim 1 to state that the object is being created "for use by" the new application instance. This change to Claim 1 is made to prevent the Examiner from confusingly stating that OOP creation of an instance of an object (as disclosed by Hapner) is same as Applicants' creation of an object which is for use by an application instance. Support for adding the word "use" to Claim 1 can be found throughout Applicants' originally-filed specification, including, for example, page 14 line 2 and page 6 line 24.

Once again, Applicants submit that the above-described clarification and the related changes to Claim 1 do not narrow Claim 1, because Applicants have merely made explicit that which was inherently present in the claim as originally filed. Therefore there is no loss of doctrine of equivalence in making these changes to Claim 1. If the Examiner believes that the scope of Claim 1 has changed, the Examiner is respectfully requested to explain their belief in the next Office Action.

Note that the term "object" as recited in Claim 1 is used in its generalized sense. Specifically, Claim 1's object has nothing to do with object oriented programming (i.e.

Claim 1 does not require nor does it preclude use of an object inheritance mechanism). This interpretation of the term “object” is apparent to a skilled artisan from the originally-filed specification, e.g. at page 6 lines 2-11.

Applicants further traverse the Examiner’s statements that Hapner discloses “said existing instance being one of the multiple instances” at column 10 lines 50-54, column 10 lines 65-67 and at column 8 lines 60-65, as stated in the middle of page 3 of the Office Action. Nothing in the Examiner cited text supports the Examiner’s position, because Hapner’s multiple instances are of an object, which fails to disclose or suggest multiple instances of a single application that contains multiple processes.

Applicants also traverse the Examiner’s statements that Hapner discloses “setting up a connectivity between the new instance and the network” at column 8 lines 40-43, column 6 lines 38-39, column 6 at line 21, and at column 11 lines 1-5 as stated in the bottom half of page 3 of the Office Action. Nothing in the Examiner cited text even remotely suggests the setting up of connectivity between an application instance and a network. Instead, all of this text by Hapner at most suggests use of pre-existing connectivity between an application instance and the network, at least because Hapner states that “ORB together with an Object Adapter (OA) is responsible ... to perform these services” (see column 8 at lines 42-43). Hence, Hapner’s connection establishment with a target object appears to be based on such pre-existing connectivity.

Moreover, Applicants traverse the Examiner’s statements that Hapner discloses “starting execution of the new instance” at column 12 lines 32-33 and at column 10 lines 2-4 as stated at the top of page 4 of the Office Action. Nothing in the Examiner-cited text by Hapner even remotely suggests the start up of an application instance. Instead, Hapner at most suggests the object is present in the address space (memory) of the “distributed object environment” and/or an “object development framework” and the object is activated, e.g. when called by a client method/function call.

As would be apparent to the skilled artisan, the startup of multiple processes of an application may be analogized, at most, to starting of Hapner’s “distributed object environment” and/or an “object development framework”. However, when activating an object, Hapner presumes that his “distributed object environment” and/or an “object

development framework” is already operational and therefore does not indicate a need to start an application instance at this stage.

In view of the numerous arguments listed above, Applicants submit that Claim 1 is not anticipated by Hapner. In order for Hapner to anticipate Claim 1, the “identical invention must be shown in as complete detail as is contained in the ...claim.” **Richardson v. Suzuki Motor Co.**, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989); MPEP § 2131. Since Hapner fails to disclose one or more limitations of Claim 1 as noted above, Applicants respectfully request the Examiner to withdraw the prior art rejection of Claim 1. Claims 2-14 and 26-28 depend from Claim 1 and are patentable for at least the same reasons as Claim 1.

The Examiner rejected the dependent claims, for several reasons that are hereby respectfully traversed, i.e. not accepted by the Applicants. Although the Examiner’s remarks regarding these claims have been rendered moot, in view of the above-described patentability of these dependent claims (for the same reasons as Claim 1), Applicants will now discuss some of these remarks to explain the irrelevance of Hapner’s teachings.

Hapner fails to disclose or suggest making a copy of an existing object, to rename the copy. The Examiner-cited text at column 7 lines 38-44 and column 10 at lines 8-20 in Hapner’s patent fails to anticipate Claim 2 for at least two reasons. First, the replica being made by Hapner is simply an in-memory copy for faster access, and not for use by another application instance. Second, Hapner’s replacement of an in-memory pointer with a persistent pointer (i.e. switching from using in-memory object to using persistent object data in a database) teaches away from changing the name of an object (i.e. only a single object is involved at this stage). Therefore, there is no suggestion whatsoever to rename a copied instance of an application as recited in Claim 2.

Furthermore, Applicants respectfully traverse the Examiner’s statement in the top half of page 5 of the Office Action that Hapner discloses where database is a file at column 4 lines 52-67 and column 5 lines 1-4. Specifically, there is no mention of “where database is a file” in the entire Hapner patent. The Examiner’s statement has no support whatsoever in Hapner’s teachings. Claim 4 is therefore patentable for this additional reason.

Applicants also traverse the Examiner's statement toward the bottom of page 5 of the Office Action that Hapner discloses Claim 14 at column 6 lines 6-15 and column 12 lines 1-10. Hapner is merely describing mapping of a persistent data object into memory. There is nothing in this cited text which even remotely suggests the addition of an entry for the new instance in a map file that identifies which instances are running on which computers. Support for the amendment to Claim 14 is found at, for example, page 14 at lines 19-21.

Claims 3, 7-9 and 12 were rejected over Hapner's teachings in view of Official Notices taken by the Examiner. Applicants respectfully traverse all of the official notices being taken by the Examiner in this Office Action. In this context, Applicants respectfully draw the Examiner's attention to the following evidentiary requirement to be met: "If Applicant Challenges a Factual Assertion as Not Properly Officially Noticed or not Properly Based Upon Common Knowledge, the Examiner Must Support the Finding With Adequate Evidence." See MPEP 2144.03. Therefore, the Examiner is hereby requested to supply a prior art reference for each "official notice."

Even assuming the Examiner supplies a prior art document that teaches one or more of the elements missing from Hapner, Applicants submit that there is no motivation or suggestion to modify Hapner's teachings. For example, the Examiner's statement that one "would have been motivated ... to complete a task by reducing or eliminating human intervention" is so generic as to be inadequate to support the Examiner-proposed modifications to Hapner's teachings. Applicants respectfully traverse this statement by the Examiner, and request the Examiner to supply a prior art reference for the motivation. Again, see MPEP 2144.03. Regarding the rejection of Claim 3, as noted above, Hapner's pointer replacement is insufficient to anticipate a name change.

Claim 13 was rejected over the teachings of Hapner in view of Callsen. However, column 4 lines 13-30 in Callsen (cited by the Examiner) makes no mention whatsoever of a "private file" contrary to the Examiner's statement at line 4 of page 8 of the Office Action. Hence, the Examiner's obviousness rejection of Claim 13 is prima facie defective. The Examiner's motivation at page 8 lines 5-6 of the Office Action is also traversed as being unsupported in prior art, and for being insufficient to make the proposed combination. The Examiner has not explained how addition of an object to a local file reduces computing

overhead, and if this is true then why isn't this motivation followed to localize everything in Hapner's patent (thereby to eliminate the network, following the Examiner's logic).

Claim 26 was also rejected over the teachings of Hapner in view of Callsen. However, column 11 lines 55-67 in Callsen (cited by the Examiner) makes no mention whatsoever of more/less powerful contrary to the Examiner's statement at line 9 of page 8 of the Office Action. Therefore, the Examiner's rejection of Claim 26 is prima facie defective. The Examiner's motivation in the middle of page 8 of the Office Action is traversed as being unsupported in prior art, and insufficient to make the combination.

Claims 10, 11 and 27 were all rejected over the teachings of Hapner in view of Snyder. Note that column 11 lines 57-65 in Snyder that were cited against Claims 10 and 11 merely say that if server installation hooks written by a programmer return FALSE, then abort installation currently in progress. Snyder makes no mention whatsoever of installing software (if such software is absent) as per Claim 10. Moreover, Snyder also makes no mention whatsoever about setting up resources as per Claim 11. Therefore, the Examiner's rejection of each of Claims 10 and 11 is prima facie defective.


Regarding Claim 27, note that at column 14, lines 13-24, Snyder is merely talking about "static member functions" of a class that can be called without creation of a class object. In this Examiner-cited text, there is no mention whatsoever of "static configuration" as recited in Claim 27. Hence, the Examiner has once again made unsupported statements, which make the rejection of Claim 27 prima facie defective.

In view of the above, Applicants believe all Claims 1-14 and 23-29 are patentable over the cited references.

Therefore, Applicants respectfully request allowance of all pending claims. Should the Examiner have any questions concerning this response, the Examiner is invited to call the undersigned at (408) 982-8203.

**Via Express Mail Label No.  
EV 581 855 672 US**

Respectfully submitted,

  
Omkar K. Suryadevara  
Attorney for Applicants  
Reg. No. 36,320

SILICON VALLEY  
PATENT GROUP LLP

10 Mission College Blvd  
Suite 360  
Santa Clara, CA 95054  
(408) 982-8200  
FAX (408) 982-8210